

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-055835

(43)Date of publication of application : 20.02.2002

(51)Int.Cl.

G06F 9/54
G06F 9/44
G06F 12/02

(21)Application number : 2000-244856

(71)Applicant : OMRON CORP

(22)Date of filing : 11.08.2000

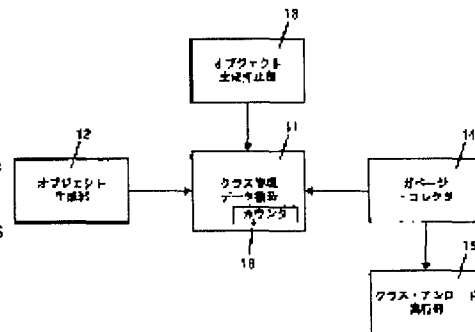
(72)Inventor : KONAKA YOSHIHARU
HIRONO MITSUAKI
SAIKI HIDEAKI
KADOWAKI MASANORI
MURAI KENICHI

(54) PROGRAM UNLOADING SYSTEM AND STORAGE MEDIUM

(57)Abstract:

PROBLEM TO BE SOLVED: To clearly unload an optional program in each prescribed unit in a program execution mechanism having structure capable of loading a necessary program in each prescribed unit such as a class in accordance with the execution of the program.

SOLUTION: When class deletion is requested, an object generation suppression part 13 sets up an object generation suppression flag in class management data structure 11 to an inhibited state. When the flag is in the inhibited state, an object generation part 12 does not newly generate an object even when object generation is requested. Thus, objects are only recovered by a garbage collector 14 and gradually reduced. At the time of detecting the deletion of all objects, a class unload execution part 15 unloads all classes. Then the collector 14 deletes a class loader object for loading the classes.



LEGAL STATUS

[Date of request for examination] 27.11.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision]

of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-55835

(P2002-55835A)

(43) 公開日 平成14年2月20日 (2002.2.20)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード* (参考)	
G 0 6 F 9/54		G 0 6 F 9/44	5 3 0 P	5 B 0 6 0
9/44	5 3 0	12/02	5 3 0 E	5 B 0 7 6
12/02	5 3 0	9/06	6 4 0 C	

審査請求 未請求 請求項の数2 OL (全 10 頁)

(21) 出願番号 特願2000-244856(P2000-244856)

(22) 出願日 平成12年8月11日 (2000.8.11)

(71) 出願人 000002945

オムロン株式会社

京都市下京区塩小路通堀川東入南不動堂町
801番地

(72) 発明者 小中 義治

京都府京都市右京区花園土堂町10番地 オ
ムロン株式会社内

(72) 発明者 廣野 光明

京都府京都市右京区花園土堂町10番地 オ
ムロン株式会社内

(74) 代理人 100094019

弁理士 中野 雅房

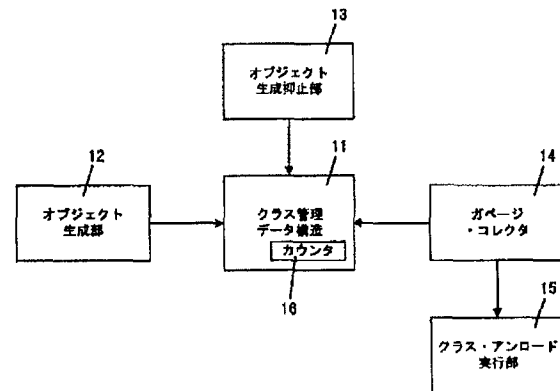
最終頁に続く

(54) 【発明の名称】 プログラムアンロードシステム及び記憶媒体

(57) 【要約】

【課題】 プログラム実行に応じて必要なプログラムをクラスなど所定単位でロードできる仕組みを有するプログラム実行機構において、任意のプログラムを所定単位で明示的にアンロードすることができるようにする。

【解決手段】 クラスを削除する要求があると、オブジェクト生成抑止部13は、クラス管理データ構造11内のオブジェクト生成抑止フラグを禁止状態にする。オブジェクト生成抑止部13が禁止状態であると、オブジェクト生成部12はオブジェクト生成を要求しても新たにオブジェクトを生成しない。このため、オブジェクトはガベージ・コレクタ14によって回収されるだけとなり、次第に減少する。クラス・アンロード実行部15は、全てのオブジェクトが削除されたことを検出すると、全てのクラスをアンロードする。続けて、ガベージ・コレクタ14は、クラスをロードしていたクラスローダー・オブジェクトを削除する。



【特許請求の範囲】

【請求項1】 プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、

所定単位のプログラムから生成されるオブジェクトの生成を抑止する機構と、所定条件下でオブジェクトを消滅させる機構と、

所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、

すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムからアンロードする機構とを有することを特徴とするプログラムアンロードシステム。

【請求項2】 プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、

所定単位のプログラムから生成されるオブジェクトの生成を抑止する機構と、所定条件下でオブジェクトを消滅させる機構と、

所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、

すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムからアンロードする機構とを有するプログラムアンロードシステムを格納した記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、プログラムアンロードシステムおよび記憶媒体に関する。特に、Javaプログラム実行環境のようなプログラム実行機構で、クラスをアンロードする処理において、明示的にクラスをアンロードできるようにするためのものである。

【0002】

【背景技術】プログラム実行に応じて必要なプログラムをクラス単位などでロードできる仕組みを持つプログラム実行機構、例えばJavaプログラム実行環境（Java言語で記述されたソースプログラムをコンパイルすることにより生成される、中間コードを実行するJava仮想マシンによって実現されるプログラム実行環境）では、プログラムをインターネットやサーバ等からダウンロードして実行できるという特徴を持つ。この特徴を利用すれば、新たなプログラムをダウンロードしてシステム（Javaプログラム実行環境搭載装置）に実装させることにより、システムを停止させることなく、システムの振る舞いや機能を変更することも可能になる。例えば、図1に示すように、不具合修正版のクラスA、機能追加（削除）版のクラスA、収集データ変更版などの新たなクラスAをシステム1にロードすることにより、不具合のあるクラスAを修正済みのクラスに置き換えたり、新たな機能を追加したり、不要な機能を除去した

り、エージェントにより収集されたデータを変更または追加することができる。

【0003】このような方式を実現するためには、システム動作中に変更または修正されたクラスを再度システムにロードする必要がある。現在のJavaプログラム実行環境では、同名のクラスを再ロードするためには、置き換えたいクラスを一旦システムから削除しなくてはならない。このようにクラスをシステムから削除することを「クラスをアンロードする」という。また、クラスとは、オブジェクト（1つのデータ構造で、データの値とそれに関する手順を合わせたものであって、クラスにより生成される。）内部のデータ構造とその操作をまとめて定義したものである。

【0004】通常のJavaプログラム実行環境においても、ロードされたクラスを再びアンロードする仕組みは組み込まれているが、通常のJavaプログラム実行環境に組み込まれているクラスをアンロードする仕組みは、次に説明するように、あるクラスから生成されたオブジェクトが全てガベージ・コレクトの対象となったとき、そのクラスをアンロードするというものである。

【0005】図2、図3及び図4は、Javaプログラム実行環境において、Java仮想マシンにクラスがロードされ、またアンロードされる仕組みを表している。例えば図2に示すクラスA1のオブジェクトが、その処理の中でクラスA2を呼び出すと、クラスA2がJava仮想マシンにロードされ、図5（a）のステップS1に示すようにオブジェクト生成部がオブジェクト3の作成を要求するので、ロードされたクラスA2はJava仮想マシン上でインスタンス化（Java仮想マシン上で実行可能な状態となること）されてオブジェクト3（インスタンスともいう。）が作成される（ステップS2）。そして、呼び出したクラスA1のオブジェクト3は、クラスA2のオブジェクト3を利用できるようになる。このようにして、Java仮想マシン上においては、図2に示すようにクラスローダー・オブジェクト2によって各クラスA1、A2、…がロードされ、さらに各クラスA1、A2、…からインスタンス化されたオブジェクト3の群が生成される。一方、Javaプログラム実行環境においては、あるオブジェクト3が他のオブジェクト3によって参照もしくは使用されていないことがガベージ・コレクタによって確認されると（図5（b）のステップS11でYESの場合）、ガベージ・コレクタは使われなくなったオブジェクト3を回収（クリア）する（ステップS12）。ついで、クラスA1、A2、…から生成されたオブジェクト3が全てガベージ・コレクタによって回収されたか否かが判別し（ステップS13）、図3に示すようにクロスローダ・オブジェクト2から派生した全てのオブジェクト3が全て回収されていれば、クラスA1、A2、…とクラスローダー・オブジェクト2をJava仮想マシンからアンロードする

(ステップS14)。なお、クラスローダーとは、Java プログラム実行環境において、クラスファイルを読み込み、クラスをメモリ上に展開するものであって、プログラム実行に必要な情報が納められている。

【0006】しかし、オブジェクト3は任意のタイミングで生成されているため、従来のガベージ・コレクタによるオブジェクト回収方式では、使用されないオブジェクト3がガベージ・コレクタによって回収される一方、図4に示すように、クラスA1、A2、…からオブジェクト3が生成され続けており、いつまでたっても全てのオブジェクト3が回収されることが無く、クラスA1、A2、…をアンロードすることができない可能性がある。このため、いつまでたってもクラスを再ロードして置き換えることができない恐れがある。

【0007】これを解決するためには、クラスに別名を付けて、ロードするという方法もある。例えば、既に組み込まれているクラスA ver1に対してクラスA ver1で置き換える代わりに、図6に示すように、クラスA ver2、クラスA ver3というように別名でロードし、別名でロードしたクラスA ver3を実行させるというものである。

【0008】しかし、このような方法では、システム内に不要なクラス(上記の例では、クラスA ver1やクラスA ver2)が残ってしまい、メモリの使用効率が悪くなる。パソコン上のJava プログラム実行環境では、図6(a)に示すように、使用するメモリ領域を拡張することができるので、メモリの使用効率についてはそれほど注意を払う必要はなく、クラスA ver1やクラスA ver2などの古いクラスを残しておいても差し支えない。これに対し、組み込み機器上のJava プログラム実行環境では、図6(b)に示すように、使用できるメモリに厳しい制約があるため、クラスA ver1やクラスA ver2などの古いクラスをいつまでもメモリ上に置いてメモリを無駄に使用することは許されない。また、パソコン上のJava プログラム実行環境でも、古いクラスは削除してメモリを効率的に使用することが望ましい。

【0009】

【発明の開示】本発明の目的とするところは、プログラム実行に応じて必要なプログラムをクラスなど所定単位でロードできる仕組みを有するプログラム実行機構において、任意のプログラムを所定単位で明示的にアンロードすることができるようにすることにある。

【0010】本発明にかかるプログラムアンロードシステムは、プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、所定単位のプログラムから生成されるオブジェクトの生成を抑制する機構と、所定条件下でオブジェクトを消滅させる機構と、所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、す

べてのオブジェクトが消滅させられた所定単位のプログラムをシステムから削除する機構とを有することを特徴としている。

【0011】また、本発明にかかる記憶媒体は、プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、所定単位のプログラムから生成されるオブジェクトの生成を抑制する機構と、所定条件下でオブジェクトを消滅させる機構と、所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムから削除する機構とを有するプログラムアンロードシステムを格納したものである。

【0012】ここで、プログラムをロードする所定単位とは、例えばJava 仮想マシンにおけるクラスであるが、これに限るものではない。また、オブジェクトを消滅させる機構とは、例えばJava 仮想マシンにおけるガベージ・コレクタであるが、これに限るものでなく、オブジェクトを消滅させる所定条件とは、例えば当該オブジェクトが使用されないことである。また、プログラムアンロードシステムを格納する記憶媒体の種類は、特に限定されるものではないが、例えばハードディスク、CD類、DVD、各種ICメモリ、MOなどが含まれる。

【0013】本発明にあつては、クラス等の所定プログラムから生成されるオブジェクトの生成を抑制する機構を備えているので、目的とするプログラムのオブジェクト生成を抑制すると、オブジェクト消滅機構によって次第にオブジェクトの数が減少させられていく。そして、所定プログラムから生成した全てのオブジェクトが消滅したことを前記検出機構により検出すると、当該プログラムをシステムから削除する。よって、本発明によれば、目的とするプログラムのオブジェクト生成を抑制させることで、明示的に目的とするプログラムを削除することができる。

【0014】

【発明の実施の形態】(第1の実施形態)図7は本発明の一実施形態によるプログラムアンロードシステムの構成を示す概略図である。このプログラムアンロードシステムは、クラス管理データ構造11、オブジェクト生成部12、オブジェクト生成抑制部13、ガベージ・コレクタ14、クラス・アンロード実行部15から構成されている。クラス管理データ構造11は、クラス及びオブジェクトを管理しており、クラスローダー・オブジェクトによってロードされた各クラスから生成されているオブジェクトの数をカウンタに保持し更新している。また、クラス管理データ構造11は、各クラスのオブジェクトの生成を許可又は禁止するためのフラグ(以下、オブジェクト生成抑制フラグという。)を保持している。

なお、このプログラムアンロードシステムの各構成要素は、主としてソフトウェアによって構成されるものであり、例えば、ハードディスク、CD-ROMやCD-RW等のCD類、DVD、MOなどの記憶媒体に格納される。

【0015】クラス管理データ構造11のオブジェクト生成抑止フラグは、通常は許可状態となっているが、オブジェクト生成抑止部13は、要求によりクラス管理データ構造11のオブジェクト生成抑止フラグを禁止状態に設定することができる。

【0016】オブジェクト生成部12は、要求に応じて各クラスからオブジェクトを生成させるものであって、オブジェクト生成抑止フラグが許可状態の場合には、要求に応じてオブジェクトを生成し、クラス管理データ構造11内に保持されている、オブジェクト数カウント用のカウンタ16を1増加させるが、オブジェクト生成抑止フラグが禁止状態の場合には、オブジェクト生成を要求されてもオブジェクトを生成しない。すなわち、図8のフロー図に示すように、オブジェクト作成要求があったとき（ステップS21）には、オブジェクト生成部12は、オブジェクト生成抑止フラグを参照して当該フラグが許可状態か禁止状態か判定する（ステップS22）。そして、クラス管理データ構造11のオブジェクト生成抑止フラグが許可状態ならば（ステップS22でNOの場合）、クラスからオブジェクトを生成し（ステップS24）、クラス管理データ構造11に保持しているオブジェクト数のカウンタ16を1増加させる（ステップS25）。オブジェクト生成抑止フラグが禁止状態ならば（ステップS22でYESの場合）、オブジェクト生成要求があってもオブジェクト生成部12は新たなオブジェクトの生成を禁止され、新たなオブジェクトを生成しない（ステップS23）。

【0017】ガベージ・コレクタ14は、使用されなくなったオブジェクトに割り当てられていたメモリ領域を自動的に解放するメモリ管理機構である。このガベージ・コレクタ14は、図9のフロー図に示すように、オブジェクトの使用状況を監視しており（ステップS31）、いずれかのオブジェクトが使用されていない場合には、当該オブジェクトを回収し（ステップS32）、クラス管理データ構造11に保持しているオブジェクト数のカウンタ16を1減少させる（ステップS33）。また、派生している全てのクラスがオブジェクトを持たなくなったクラスローダー・オブジェクトは、ガベージ・コレクタ14による回収の対象となる。

【0018】クラス・アンロード実行部15は、クラス管理データ構造11に保持しているオブジェクト数を参照し、オブジェクトを生成していないクラスをアンロードするものである。すなわち、図10のフロー図に示すように、クラスアンロードの実行を開始する指示があった場合には（ステップS41）、オブジェクト生成抑止

部13は、クラス管理データ構造11内のオブジェクト生成抑止フラグを禁止状態に設定する（ステップS42）。一方、クラス・アンロード実行部15は、クラスローダー・オブジェクトによって生成された全てのクラスのオブジェクトを監視しており（ステップS43）、全てのクラスのオブジェクトが回収されてクラス管理データ構造11内のオブジェクト数のカウンタ16がゼロになると、全てのクラスをシステムからアンロードする（ステップS44）。これによって、クラスローダー・オブジェクトによって読み込まれ、メモリ上に展開されたクラスが削除される。

【0019】図11（a）（b）及び図12（c）

（d）（e）は、上記のような構成のプログラムアンロードシステムにより、クラスA1、A2、…及びオブジェクト3が生成されてから、完全に削除されるまでの様子を模式的に表している。アプリケーションプログラムが実行されると、図11（a）に示すように、Java仮想マシン内にクラスローダー・オブジェクト2が生成され、クラスローダー・オブジェクト2によってプログラム実行に必要なクラスA1、A2、…がロードされる。さらに、各クラスA1、A2、…からは、オブジェクト生成部12により任意にオブジェクト3が生成される。一方、使用されなくなったオブジェクト3は、ガベージ・コレクタ14によって回収されている。従って、クラス管理データ構造11内のオブジェクト生成抑止フラグが許可状態になっている場合には、常にオブジェクト3は生成と消滅を繰り返している。

【0020】これに対し、例えば新規のクラスをロードするために実行中のクラスを削除する必要がある場合には、アンロード指示を受けてオブジェクト生成抑止部13がオブジェクト生成抑止フラグを禁止状態に設定する。当該フラグが禁止状態になると、図11（b）に示すように、それ以降オブジェクト生成部12によって各クラスA1、A2、…からオブジェクト3が生成されなくなるので、使用されないオブジェクト3がガベージ・コレクタ14によって回収されることにより、次第にオブジェクト数は減少していく。

【0021】クラスローダー・オブジェクト2によりロードされた全てのクラスA1、A2、…から生成しているオブジェクト3がすべてガベージ・コレクタ14によって回収され、図12（c）のようにクラスA1、A2、…から生成したオブジェクト3が存在しなくなると、クラスローダー・オブジェクト2もガベージ・コレクタ14による回収の対象となる。オブジェクト3がすべてガベージ・コレクタ14によって回収されたかどうかは、クラス管理データ構造11に設けられているオブジェクト数のカウンタ16がゼロになることで判断される。クラスローダー・オブジェクト2がガベージ・コレクタ14による回収の対象となり、カウンタ16がゼロになると、図12（c）に示すように、クラス・アンロ

10

20

30

40

50

ード実行部 15 によって全てのクラス A1、A2、…のアンロードが開始され、クラス A1、A2、…がすべて削除されると、クラスローダー・オブジェクト 2 もガベージ・コレクタ 14 によって削除される。

【0022】この結果、Java 仮想マシンに新しいクラスをロードすることができるようになる。

【0023】(第2の実施形態) 上記実施形態では、クラスローダー・オブジェクト 2 によってロードされた全てのクラスから生成されたオブジェクト数をクラス管理データ構造 11 のカウンタ 16 で数えるようにしていたが、オブジェクト数を数える代わりに、生成されているオブジェクトのリストをクラス管理データ構造 11 内に保持させるようにしてもよい。すなわち、この実施形態では、図 13 に示すように、オブジェクト生成部 12 でオブジェクトを生成させた場合には、当該オブジェクトをクラス管理データ構造 11 のリストに追加する(ステップ S25a)。また、ガベージ・コレクタ 14 によってオブジェクトを削除した場合には、クラス管理データ構造 11 のリストから当該オブジェクトを削除する(ステップ S33a)。また、リストにオブジェクトが登録されていないと判断したとき(ステップ S43a)、クラス・アンロード実行部 15 によって全てのクラスをシステムから削除する(ステップ S44a)。

【0024】このような実施形態でも、第1の実施形態と同様にしてクラスを明示的にアンロードすることが可能になる。

【0025】

【発明の効果】本発明によれば、目的とするプログラムのオブジェクト生成を抑止させることで、明示的に目的とするプログラムを削除することができる。よって、確実にシステムに新たなプログラムを再ロードすることができる。

【図面の簡単な説明】

【図1】システムにロードされているクラスを再ロードする必要のある状況を説明する図である。

【図2】従来の Java プログラミング実行環境において、クラスローダー・オブジェクトによりクラスがロードされ、各クラスからオブジェクトが生成している様子を説明する図である。

【図3】同上の Java プログラミング実行環境において、全てのオブジェクトが回収(削除)された状態を説明する図である。

【図4】同上の Java プログラミング実行環境において、新しいオブジェクトが生成している様子を説明する図である。

【図5】(a)は同上の Java プログラミング実行環境における、オブジェクト生成部の働きを示すフロー図、(b)はガベージ・コレクタの働きを示すフロー図である。

【図6】(a)はクラスをロードされたパソコン上の Java プログラミング実行環境におけるメモリ領域使用状況を示す図、(b)はクラスをロードされた組込み機器上の Java プログラミング実行環境におけるメモリ領域使用状況を示す図である。

【図7】本発明の一実施形態によるプログラムアンロードシステムの構成を示す図である。

【図8】同上のプログラムアンロードシステムに用いられているオブジェクト生成部の動作を説明するためのフロー図である。

【図9】図7のプログラムアンロードシステムに用いられているガベージ・コレクタの動作を説明するためのフロー図である。

【図10】図7のプログラムアンロードシステムに用いられているオブジェクト生成抑止部とクラス・アンロード実行部の動作を説明するためのフロー図である。

【図11】(a)(b)は、クラスがロードされ、各クラスからオブジェクトが生成してからクラス及びクラスローダー・オブジェクトが削除されるまでの過程を説明する図である。

【図12】(c)、(d)及び(e)は、図11(a)(b)の続図である。

【図13】本発明の別な実施形態によるプログラムアンロードシステムに用いられているオブジェクト生成部の動作を説明するためのフロー図である。

【図14】同上の実施形態によるプログラムアンロードシステムに用いられているガベージ・コレクタの動作を説明するためのフロー図である。

【図15】同上の実施形態によるプログラムアンロードシステムに用いられているオブジェクト生成抑止部とクラス・アンロード実行部の動作を説明するためのフロー図である。

【符号の説明】

2 クラスローダー・オブジェクト

3 オブジェクト

A、A1、A2、… クラス

11 クラス管理データ構造

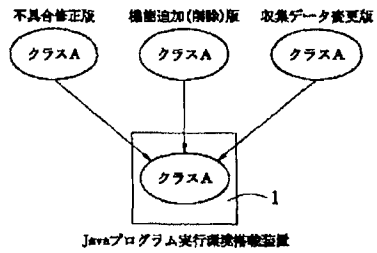
12 オブジェクト生成部

13 オブジェクト生成抑止部

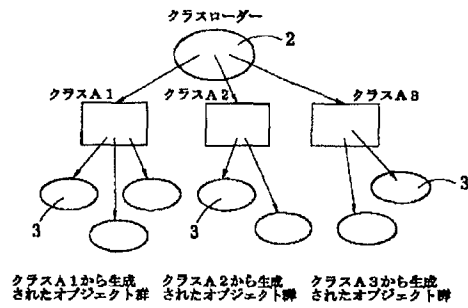
14 ガベージ・コレクタ

15 クラス・アンロード実行部

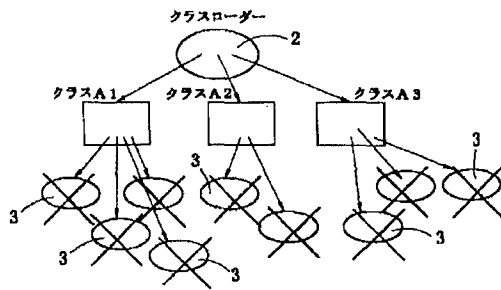
【図1】



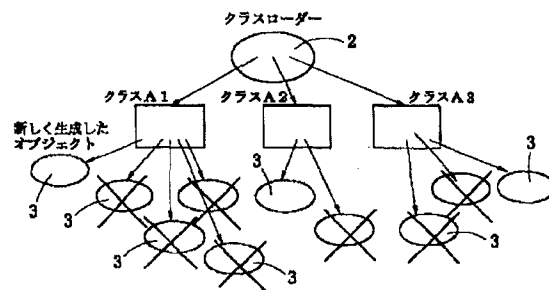
【図2】



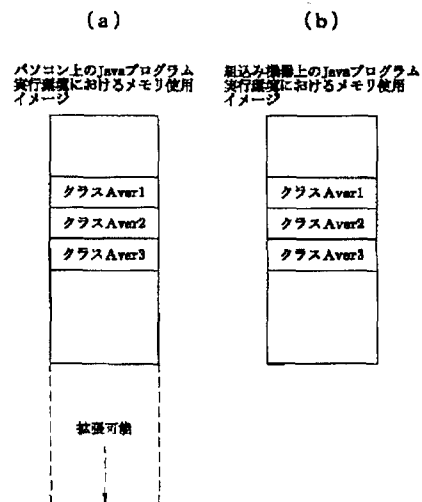
【図3】



【図4】

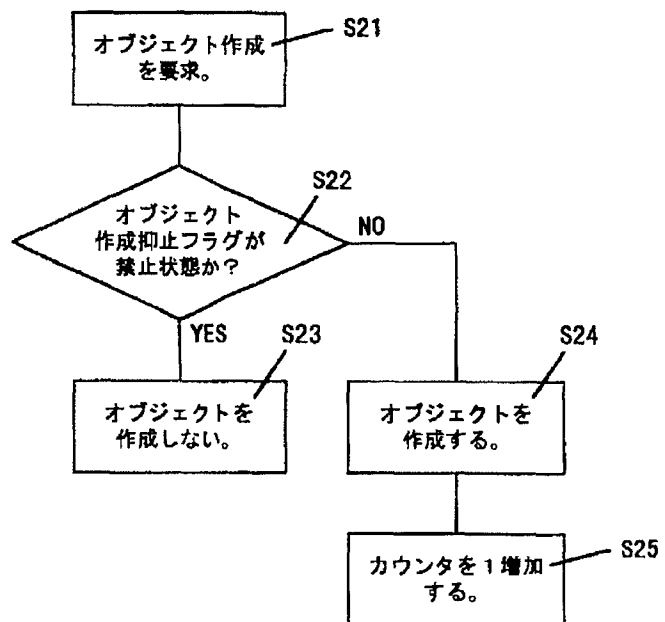


【図6】

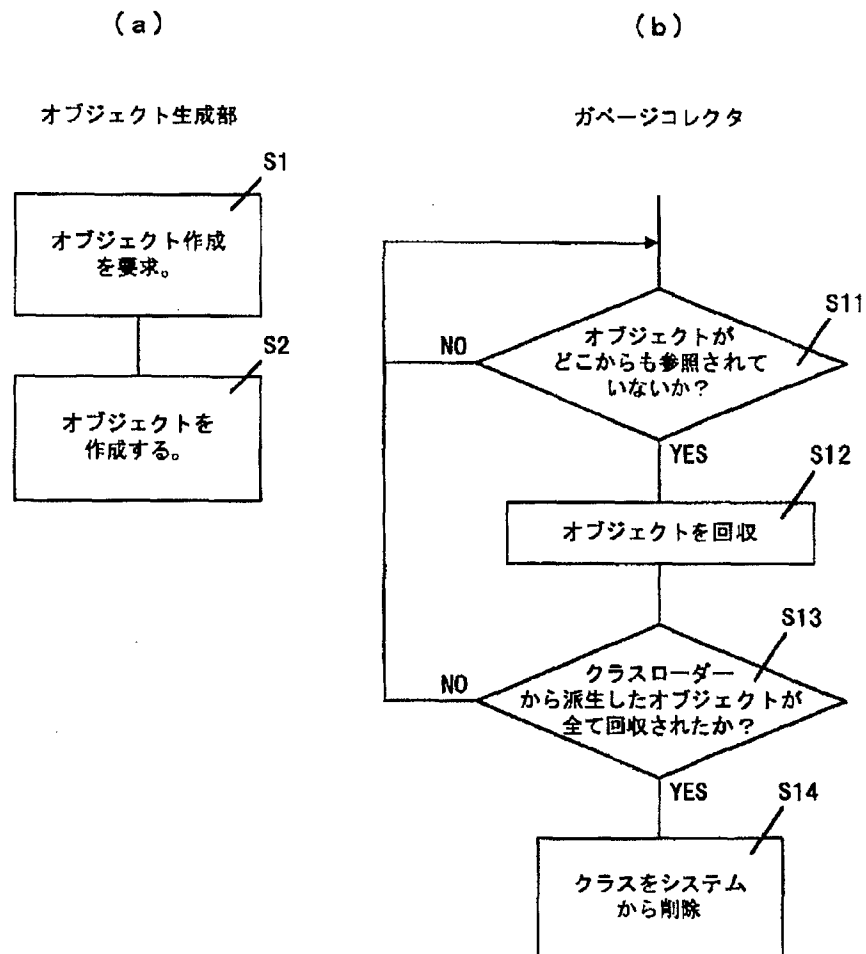


【図8】

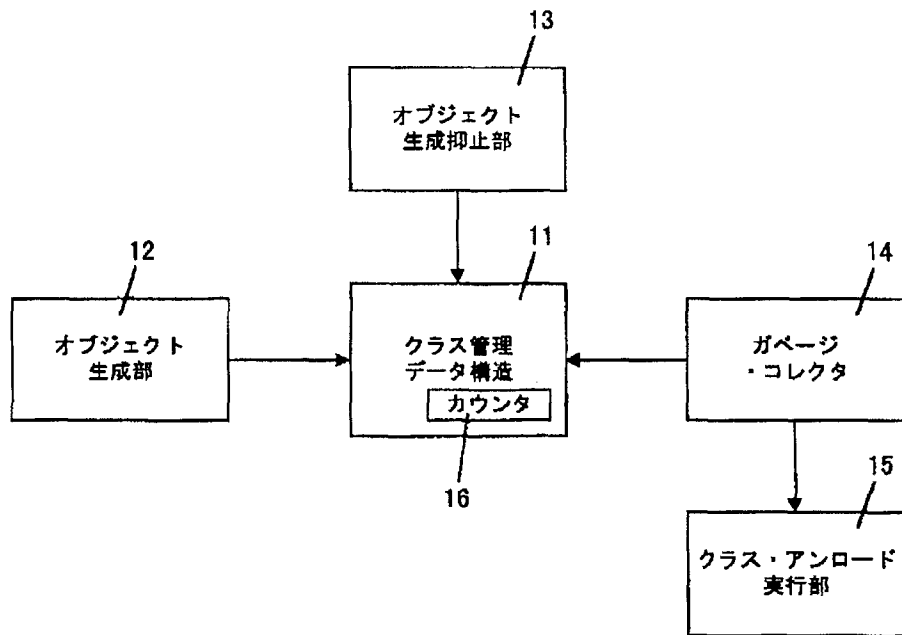
オブジェクト生成部



【図 5】

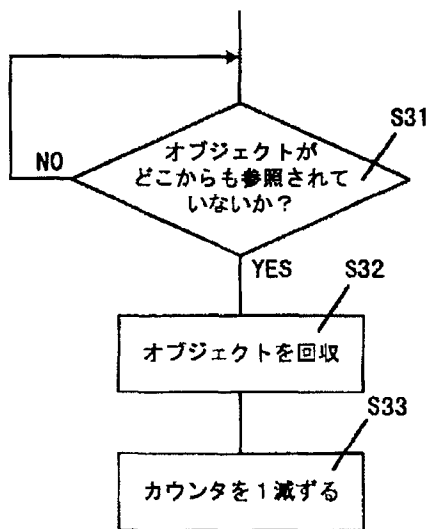


【図 7】

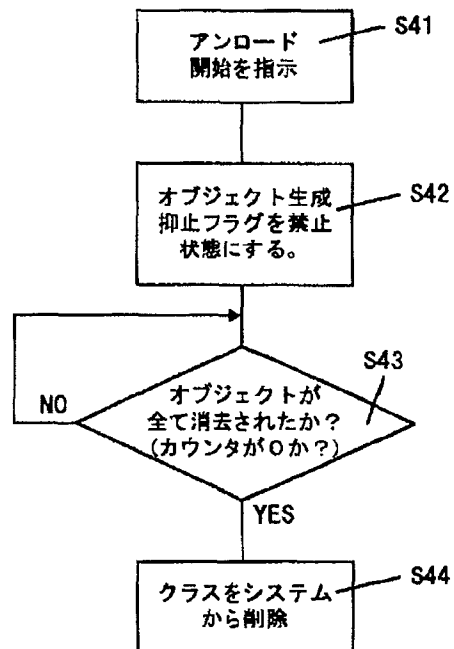


【図 9】

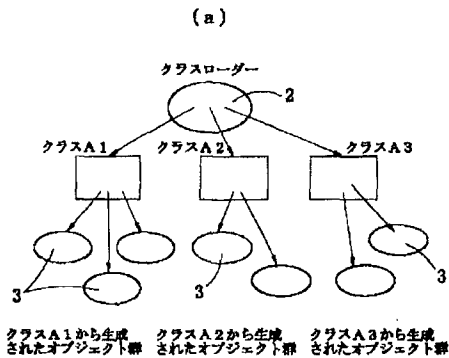
ガベージ・コレクタ



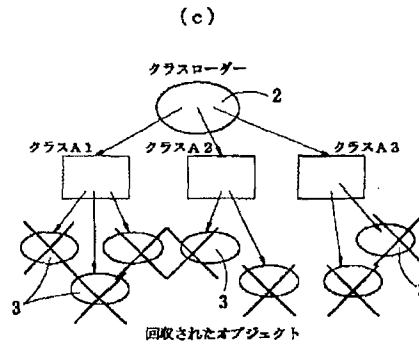
【図 10】

オブジェクト生成抑止部
クラス・アンロード実行部

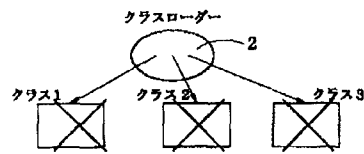
【図 11】



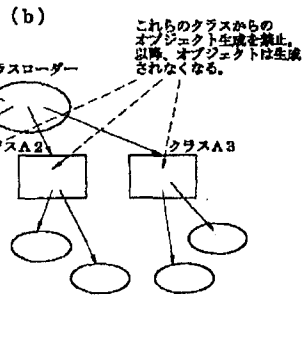
【図 12】



(d)

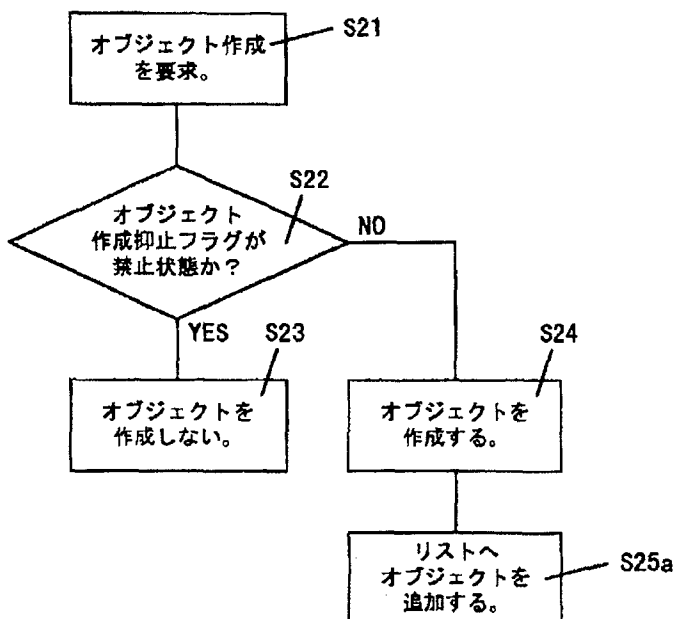


(e)



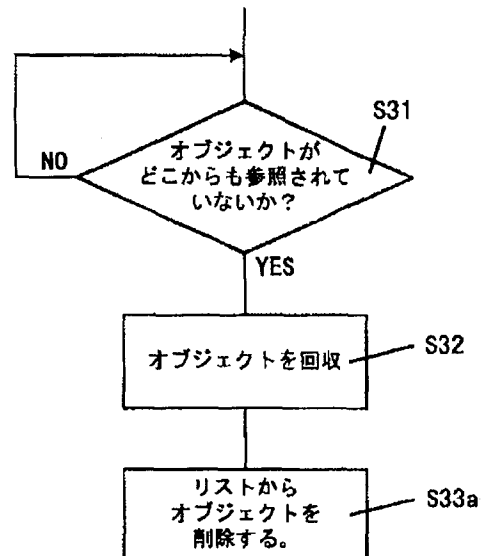
【図 13】

オブジェクト生成部



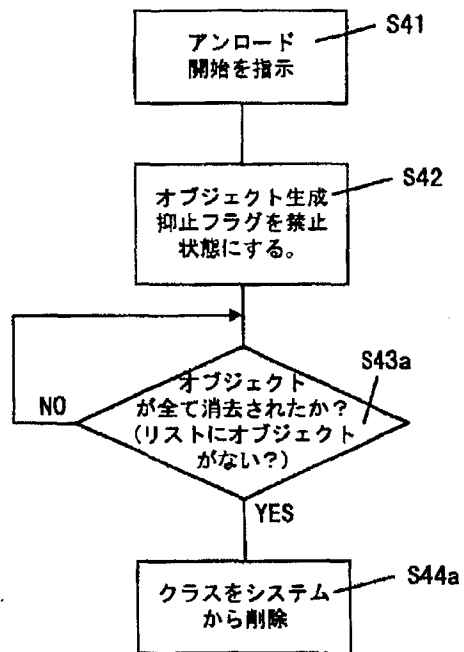
【図 14】

ガベージ・コレクタ



【図 15】

オブジェクト生成抑止部
クラス・アンロード実行部



フロントページの続き

(72)発明者 齋木 秀明

京都府京都市下京区西洞院木津屋橋通東入
ル オムロンソフトウェア株式会社内

(72)発明者 門脇 正規

京都府京都市右京区花園土堂町10番地 オ
ムロン株式会社内

(72)発明者 村井 謙一

京都府京都市右京区花園土堂町10番地 オ
ムロン株式会社内

Fターム(参考) 5B060 AA09 AA18 AC01

5B076 BA01 BA05